AT&T
Bell Laboratories

OVERVIEW OF THE AT&T's 32-BIT HOBBIT MICROPROCESSOR

The advent of micron and sub-micron IC processing technologies has brought an explosion of new microprocessor architectures which attempt to integrate a high degree of systems resources on a single chip or multiple chip set. The major thrust of these architectures has thus far been towards the development of high end work stations and mini-supercomputers, typically employing large, power hungry CPU chips surrounded by large cache chips and coprocessors and requiring staff years of software development to attain high performance. This has lead to a virtual vacuum of high performance chips aimed at low end machines which might eventually find their way into the hands of a large market. The AT&T HOBBIT microprocessor has been designed to fill this neglected need. With a die size of less than one square centimeter, AT&T HOBBIT provides high CPU performance on an easily programmed chip while dissipating slightly more than half a Watt at 3.3V operation and less than 100μW in standby mode.

The architecture of AT&T HOBBIT is a second generation implementation of the CRISP architecture originally developed at AT&T Bell Laboratories. CRISP broke the ground for current RISC microprocessors, being one of the first architectures to efficiently combine large on-chip caching with instruction pipelining to attain single cycle instruction execution. The goal of the CRISP architecture is to execute a rationalized (rather than "reduced") instruction set, optimized for high-level language implementation, on a single high-performance chip. The resulting simplicity in software design provides a distinct advantage when developing and supporting custom software environments with special purpose interpreters and compilers. Traditional RISC microprocessors have shifted the responsibility for obtaining high performance onto system software development; the CRISP architecture provides a balance by combining the salient features of both RISC and CISC machines to reduce system hardware and software complexity without sacrificing system performance.

The resulting AT&T HOBBIT microprocessor is the culmination of years of research on computer architecture, software design and processing technology. A major emphasis while developing AT&T HOBBIT was the integration of as many hardware system resources on chip as possible while retaining a small die size and thus low chip cost. The performance is scalable so that by altering the size of on-chip caches, microprocessors of varying price/performance ratios can be produced while retaining software compatibility.

AT&T - PROPRIETARY
Use pursuant to Company Instructions

The RISC-like features of Hobbit which provide powerful CPU performance include:

- A reduced instruction set retaining the most commonly used instructions and addressing modes.
- Single-cycle instruction execution for most instructions, with a peak rate of better than one cycle per instruction due to folded branch.
- A large on-chip register set with 64 32-bit "registers" implemented as a stack cache.

Some of the non-RISC features which simplify efficient Hobbit system software development include:

- Variable length instruction format for high code density.
- Memory-to-memory architecture.
- A variable sized process register frame.
- No delayed branching.
- No load delay slot.
- Hardware implemented complex instructions such as integer multiply, divide and remainder.

Other distinguishing features of Hobbit are:

- An optimized instruction set for supporting C and other high level languages.
- Tagged arithmetic support for object-oriented languages.
- Atomic instructions for semaphore support.
- A 3 Kbyte, 3-way set associative, instruction cache.
- A 32 entry, direct mapped decoded instruction cache which provides efficient execution of tight software loops.
- A prefetch decode unit which works independently of the execution pipeline.
- A three stage execution pipeline providing single cycle per instruction performance.
- A 256 byte stack cache which holds the top of the user stack on-chip.
- Hazard detection to ensure operand integrity and bypassing for peak performance.
- Branch prediction and branch folding to minimize control transfer penalties.
- User/kernel execution modes with privileged instructions for system security.
- Virtual memory support with paged and non-paged segments.
- An integrated Memory Management Unit (MMU), with two, 32-entry Translation Look-aside Buffers (TLB's), one for text and one for data.
- Instruction and block tracing capability.
- 4 Gbyte address space with minimum one cycle access synchronous I/O protocol, automatic wait state insertion and burst mode access.
- Six levels of maskable interrupts and 1 non-maskable interrupt with auto vectoring.
- Two timers for counting clock cycles, completed instructions or other events.
- An IEEE 1149.1/05 compatible Test Access Port.
- 20 MHz (3.3 V) to 33 MHz (5.0 V) worst case operation, with selection for higher frequencies.
- Low power operation (0.6 W at 3.3 V and 20 MHz) and extremely low power standby mode (under 100µW at 3.3 V).
- Ceramic 125 pin grid array, or 132 pin plastic quad flat pack (PQFP) package.

AT&T - PROPRIETARY
Use pursuant to Company Instructions

AT&T
Bell Laboratories

AT&T'S 32-BIT HOBBIT MICROPROCESSOR TECHNICAL DATA SUMMARY

The AT&T Hobbit microprocessor is a second generation, full 32-bit microprocessor which integrates both on chip caching and instruction pipelining to provide single cycle execution for most instructions. A rationalized instruction set simplifies systems programming while a high degree of system resources integrated on chip simplifies hardware development; a typical AT&T Hobbit system configuration is depicted in Figure 1. The processor has been designed to provide high CPU performance in applications requiring a small die size and low power dissipation. The chip measures less than one square centimeter in area while dissipating slightly more than half a Watt at 20 MHz, 3.3V operation.

A block diagram of the AT&T Hobbit architecture is illustrated in Figure 2. It is based on the CRISP architecture, references to which are listed at the end of this document. Here we intended to discuss some of the salient features of the architecture which provide AT&T Hobbit with a unique position among RISC architectures. The instruction set, typical instruction execution times, electrical and mechanical specifications are all summarized near the end of this document. For details of the AT&T Hobbit microprocessor not elaborated here, refer to the AT&T Hobbit Requirements Document.

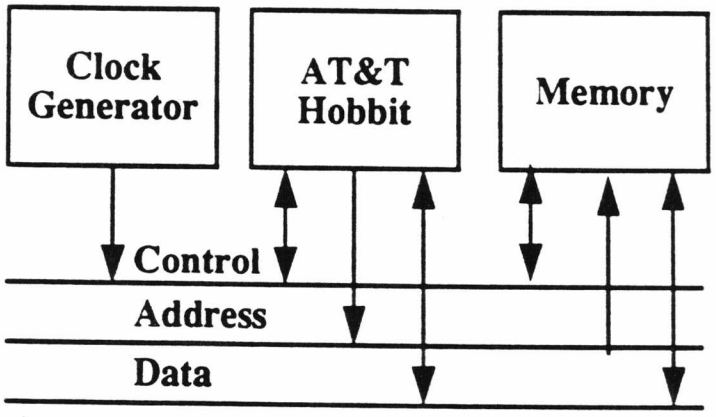


Figure 1: Block diagram of a basic AT&T Hobbit system.

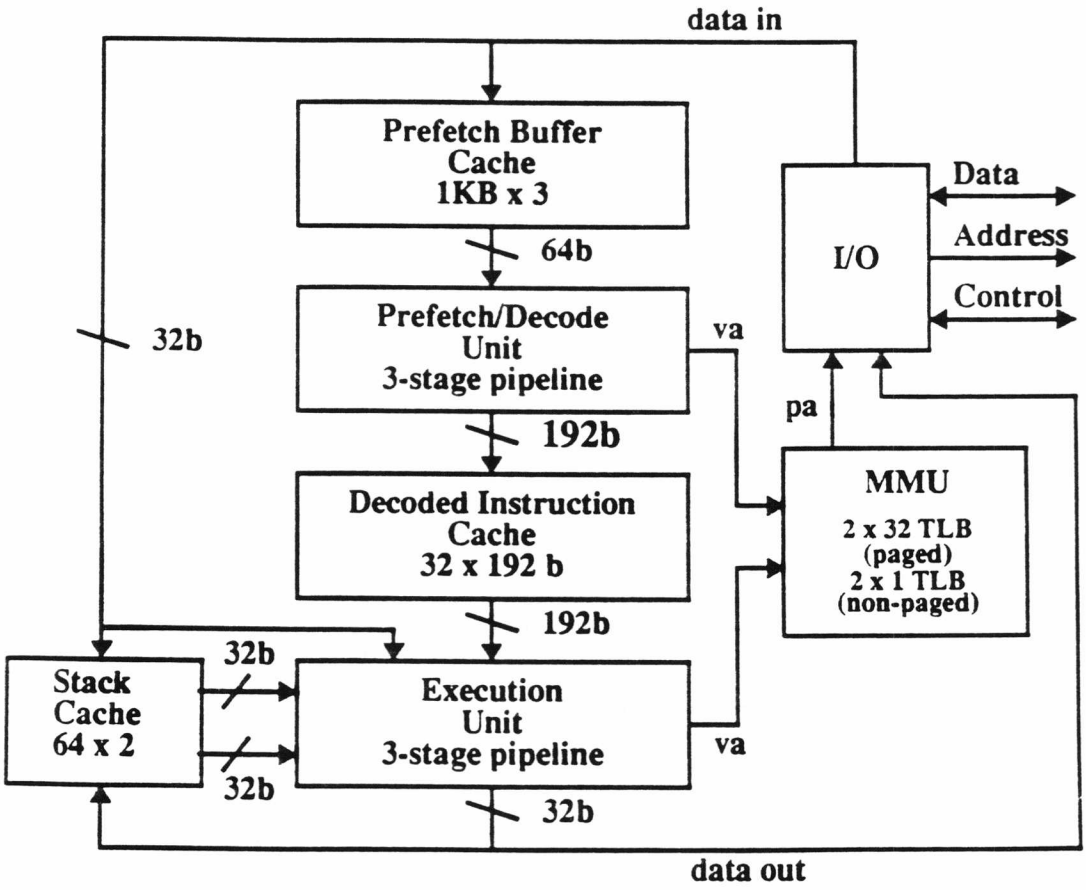
AT&T - PROPRIETARY
Use pursuant to Company Instructions

Memory-to-Memory Architecture and Addressing Modes

AT&T Hobbit operations are memory-to-memory rather than register-to-register as for conventional RISC microprocessors. Operating directly on memory contents requires fewer instructions for a given task than a load-store architecture, thereby saving code space and I/O bandwidth. Most of the AT&T Hobbit instructions are dyadic in nature with the second operand acting as an implicit destination. Several pseudo-triadic instructions are provided with the top of the stack acting as the implied destination. The non-branch instructions allow different combinations of immediate, memory (absolute), stack offset and stack offset indirect addressing modes for the two operands (Table 1a). The branch instructions can use absolute, absolute indirect, PC-relative or stack offset indirect addressing modes (Table 1b).

AT&T Hobbit does not support general purpose registers; rather, the top 256 bytes of the user stack frame is cached on-chip. The stack offset and stack offset indirect addressing modes directly manipulate the contents of this cache and thus make it available as a "register file." Even though AT&T Hobbit utilizes a memory-to-memory architecture, most stack

Figure 2: AT&T Hobbit Block diagram
(va: virtual address; pa: physical address)



AT&T - PROPRIETARY
Use pursuant to Company Instructions

references are interpreted by the processor as "register" references which hit in this stack cache.

Operand Types and Access

AT&T Hobbit supports byte addressing with signed and unsigned, byte, half-word and signed word data types. Most AT&T Hobbit instructions allow all of these operand types to be mixed with any of the four addressing modes. This symmetric operand access space greatly simplifies system software development and compiler technology.
Byte ordering within words is "big endian," *i.e.*, within a word the smallest byte address refers to the most significant byte. Non-word operands are properly aligned and sign extended to 32-bits prior to execution. All operands are fetched or stored as full 32-bit words with byte marks selecting the appropriate bytes during I/O access.

Multiple Length Instruction Format

The AT&T Hobbit instruction set (summarized in Tables 2a - 2i) is encoded in 2-byte, 6-byte, and 10-byte formats (see Figure 3). The 6-byte and 10-byte formats allow for 16-bit and full 32-bit operand addressing but typically the 2-byte format accounts for 80% of all executed code. Since two 16-bit instructions can be fetched over the 32-bit data bus, text I/O bus traffic is greatly reduced.

The encoding compactness, coupled with the non-LOAD/STORE memory-to-memory architecture, leads up to 50% higher code density over conventional register-to-register RISC microprocessors. Most of the 6-byte instructions which utilize 16-bit operands have a corresponding 10-byte instruction with 32-bit operands, so that code with absolute memory references to the low 64 Kbytes of memory is compact.

Virtual Memory Support

An integrated memory management unit (MMU) may be used to perform virtual memory translations with 4 Gbytes of virtual memory space available. The MMU supports either fixed size paged segments of 4 Kbytes each or variable size non-paged segments of 4 Kbytes up to 16 Mbytes in length. In addition to a valid bit, kernel, writable, modified and referenced bits are provided in the page tables.

The most recently translated virtual addresses for paged segments are stored in two separate 32-entry translation look-aside buffers (TLB's), one for text and one for data. In addition, two segment look-aside buffers (SLB's) are provided for non-paged segment translations. The TLB's are implemented as fully associative memories. The replacement strategy is round-robin so that the entries are written sequentially.

When virtual addressing is enabled, text and data addresses which hit in the TLB are effectively translated in zero cycles due to address prediction logic; TLB misses are automatically translated by hardware accessing the translation tables in main memory. Invalid pages and access violations generate an MMU exception. The TLBs are controlled using special instructions for invalidating selective entries or the entire TLB.

Stack Cache

One of the unique features providing AT&T Hobbit with powerful performance is the 256 byte Stack Cache, integrated on-chip. Simple stack management instructions maintain the top of the processor stack in this cache. Two CPU registers, the stack pointer (SP) and the maximum stack pointer (MSP), determine the amount of stack cached on-chip. Stack space is allocated downwards in multiples of 16 bytes.

SP and MSP are modified and any necessary stack flushing is automatically performed whenever stack space is allocated or deallocated.

The stack is accessed using either the stack offset (an offset from SP) or memory addressing modes. On-chip range check circuitry automatically determines if operand references hit in the cache. The word at SP+4, by convention called the accumulator, normally resides in the cache and serves as a destination for triadic instructions. The stack cache is implemented as a dual-ported memory so that two operands can be fetched and one stored in a single cycle.

AT&T Hobbit's Stack Cache is analogous to the register files found on other RISC processors, and its function obviates the need for such a file. Furthermore, the register frame available to a given process is up to 64 32-bit words. The stack cache size is transparent to the instruction set so that its size in future generations can be altered to improve performance *without* the need of modify existing software.

Fast Function Calling and Linkage

The Stack Cache provides a very simple but efficient subroutine linkage mechanism. The outgoing operands are stored on the stack by the calling function. On a subroutine call, the return address is automatically saved on the top of the stack (at SP). The subroutine then merely allocates its own stack frame for automatic variables, executes the code and returns to the calling procedure, usually without ever having to access off-chip stack. The return automatically deallocates the stack. The call instruction and the allocation of the stack frame can each be accomplished in a minimum of a single clock cycle.

If the called function overruns the on-chip stack, the cache is automatically flushed so that only the top 256-bytes remain on-chip. A special instruction is available for refilling the cache on the return which becomes a nop if no cache filling is required. Flushing and restoring the stack cache is very efficient due to a burst I/O feature available on AT&T Hobbit.

Pipeline

AT&T Hobbit consists of two main functional units: the prefetch decode unit (PDU) and the execution unit (EU). The PDU has three pipeline stages and is responsible for the prefetching and decoding of instructions. There are two modes of PDU operation: aggressive and passive. The EU also has three pipeline stages. The PDU and EU function almost independently of each other as described below.

Instruction Prefetch and Decode

The PDU keeps track of addresses from which the EU may request an instruction. The first stage of the pipeline, the prefetch buffer register stage (PBR) fetches the instructions either from the on-chip instruction cache (prefetch buffer, PBUF) or from off-chip memory and queues it for decoding. These instructions are then decoded in the prefetch decode register stage (PDR). The decoded instructions are provided to the EU from the prefetch instruction register stage (PIR) and are also written in the decoded instruction cache (IC).

In aggressive prefetching mode (selected through the configuration CPU register), instructions are prefetched and decoded independent of execution; if a branch is encountered a special prediction bit in the opcode allows prefetching to continue along the predicted path of execution. In the passive prefetching mode, the PDU fetches and decodes instructions only when they are requested by the execution unit, thereby reducing the I/O bus traffic.

The text is stored in PBUF which is a 3Kbyte 3-way set associative cache. The cache uses

physical tags so that it needs to be flushed only when the external memory holding text changes.The PDU utilizes a single entry "micro"-TLB which holds the physical translation for the current virtual text page; thus, when virtual address translation is enabled, text addresses need only be translated when a page boundary is crossed. A special instruction allows the user to flush individual lines of the text cache to support self-modifying code. Decoded instructions are stored in a 32-entry, direct-mapped cache which uses either physical or virtual tags depending upon how the text was fetched. The sizes of both of these caches are transparent to the user.

The Execution Unit Pipeline

The execution unit (EU) pipeline also consists of 3 stages. The Instruction Register stage (IR) resolves indirections and performs operand requests; the Operand Register stage (OR) acquires the operands and computes the result of the arithmetic operation; finally, the Result Register stage (RR) stores the result.

A peak instruction execution rate of one instruction per cycle (*i. e.* 1 CPI) can be achieved due to this pipelining. A feature called branch folding (see below) actually permits a peak rate of better than 1 CPI for loops that fit completely within the decoded instruction cache and which fetch/store operands from/to the Stack Cache. The EU is decoupled from the I/O so that once a result for an off-chip store is copied to the I/O unit, the EU pipeline can continue with other instructions that do not need I/O. Some of the multiple cycle instructions such as multiply and divide use both the OR and RR stages repetitively.

Operand Bypass and Hazard Detection

AT&T Hobbit utilizes hazard detection circuitry to compare the operand addresses in the IR stage with destination addresses in the OR and RR stages. In cases where both the addresses and data types match, the result is automatically bypassed either to the IR or OR stage. In cases where the addresses match but the data types differ, a hazard is flagged and the IR is stalled until the necessary store is complete.

Hardware hazard detection obviates the compiler's/programmer's responsibility to detect and eliminate operand hazards. Optimization to eliminate hazards may improve efficiency but is not necessary for correct execution. Hardware for bypassing in the case of hazards can improve performance; e.g. with read cancelling where off-chip memory writes are immediately followed by a read to the same location, since the read instruction will not access the I/O.

Branch Prediction and Branch Folding

The conditional branch instructions on AT&T Hobbit are encoded with a special prediction bit to indicate if the branch is likely to be taken. This bit is used by the prefetching and execution pipelines in order to reduce the penalty for branching execution flow without the need for cumbersome delayed branching techniques. The direction of most program branches can be correctly predicted at compile time. Even if a branch is miss-predicted, by arranging to modify the condition flag at least three instructions prior to the branch (to account for the three stages of the pipeline), no penalty will be incurred if the branch hits in the decoded instruction cache.

Each decoded instruction contains both a "next PC" and an "alternate next PC" field so that every instruction may potentially branch to an alternate location in text memory. When an actual branch is encountered, the target is "folded" into the alternate or next PC fields of the preceding instruction, thus effectively executing the branch in parallel. As previously pointed out, this instruction folding allows AT&T Hobbit to achieve better than 1 CPI in small repeti-

tive code segments.

Two modes of execution

AT&T Hobbit supports both a user and a kernel mode of execution enabling a secure system environment. The user may only enter the more privileged kernel mode by executing a special kernel call instruction. CPU registers (summarized in Table 3) may only be modified in kernel mode. The MMU also supports protection of the kernel virtual address space through the use of special bits in the segment and page tables. Both a privileged kernel return and a privileged context return instruction are included in the instruction set to facilitate returning from a kernel routine or event handler.

Event Processing

The interrupt stack pointer (ISP) is one of the CPU registers on the processor, and stack offset addressing modes use the ISP when the latter is enabled in the PSW. When an event such as a page-fault, interrupt or kernel call is encountered, AT&T Hobbit stores the current process information (the PC, PSW and possibly an event identifier) on the interrupt stack. The processor enters kernel mode with interrupts disabled, and program control is transferred to the appropriate event handler through a vector table (Table 4). The kernel return instruction restores the process information stored on the interrupt stack and continues program execution; the context return instruction performs the same function but also restores the SP, MSP and the Stack Cache contents as well.

Processor exceptions are generated under the following conditions: divide by zero, instruction/block tracing, address alignment fault, privilege violation (an attempt to execute a privileged instruction or writing a CPU register in user mode), unimplemented CPU register reference, instruction fetch fault, data read fault, data write fault, and I/O bus-error. Two fast unimplemented instruction traps are provided, one for unimplemented niladic instructions and the other for monadic and diadic unimplemented instructions. Each trap stores the PC on the stack and branches to an unimplemented instruction vector without changing privilege modes.

Special Features

AT&T Hobbit also includes several features not found on many other microprocessors.

- Tagged ADD/SUB instructions are provided for object oriented language support.
- Three interlocked instructions, ADDI, ANDI, and ORI, are also included which lock the I/O bus during a read-modify-write; these instructions facilitate atomic operations such as mutual code exclusion and semaphore operations.
- Floating point opcodes are defined and reserved; these generate an unimplemented instruction trap in the present implementation to simplify software floating point emulation.
- Double/quad word block access instructions using all the addressing modes are provided for efficient movement of data operands.
- Instructions are provided for flushing the prefetch cache and the decoded instruction cache.
- Two timer CPU registers are available which count either cycles or completed instructions in either user/kernel or user only mode. Both the timers can be individually configured to generate an interrupt when incrementing through zero.
- The prefetch cache, the decoded instruction cache, and the stack cache can be independently enabled/disabled through a CPU configuration register.
- An IEEE 1149.1/D5 compatible Test Access Port (TAP) is provided for boundary scan and support of the testability features. On the present implementation, all of the on-chip memories except for the two TLB's and the tags for the decoded instruction cache, incorporate build-in self-test (BIST) circuits which are

accessible through the TAP.

I/O Protocol

AT&T Hobbit supports a synchronous, minimum single cycle access I/O bus protocol. Wait states are synchronously inserted via bus control signals. The address bus is 30-bits wide with four byte marks provided for the selection of bytes within words. Bus cycles may be terminated in error, or retried. An external bus arbiter is required for multiple bus master applications.

During the read operation when the memory system tri-states after providing the data, AT&T Hobbit drives the data back on the data bus thereby preventing the data bus from floating. Low power system design without data bus pull-up resistors is thus possible.

Due to the high degree of cache integration (the text and stack caches, as well as the TLB's), AT&T Hobbit generates relatively low bus traffic. This coupled with an efficient I/O protocol enables multiple AT&T Hobbits to share the same I/O bus.

Electrical Features

AT&T Hobbit is packaged in a 125 pin grid array, or 132 pin JEDEC plastic quad flat pack (PQFP). The Table 5 summarizes the pinout. AT&T Hobbit operates with the supply voltage in the range of 3.3 V to 5.0 V ±5% (Table 6), with CMOS voltage levels for the I/O. The power dissipation at 3.3 V and 20 MHz is 0.6 W and at 5.0 V and 33 MHz is 2.2 W. AT&T Hobbit requires two quadrature clock inputs, *i.e.* one shifted 90^{o} in phase with respect to the other (Table 7). AT&T Hobbit is designed with virtually all static circuitry; a low power DC mode is supported through indefinite extension of one of the input clock edges. In this stand-by mode, the power dissipation is less than 100 μW at 3.3 V. The absolute maximum ratings are summarized in Table 8.

Figure 3: Instruction Formats

Figure 3a: 2-byte instruction formats

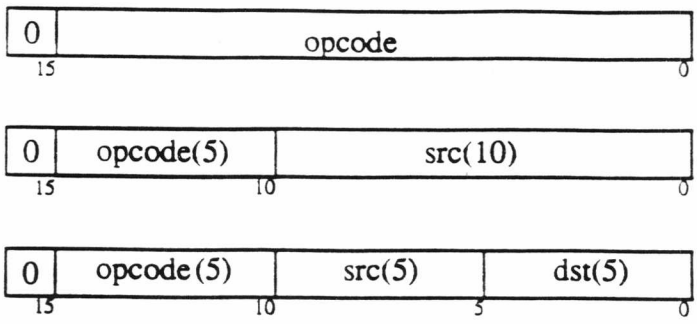


Figure 3b: 6-byte instruction formats

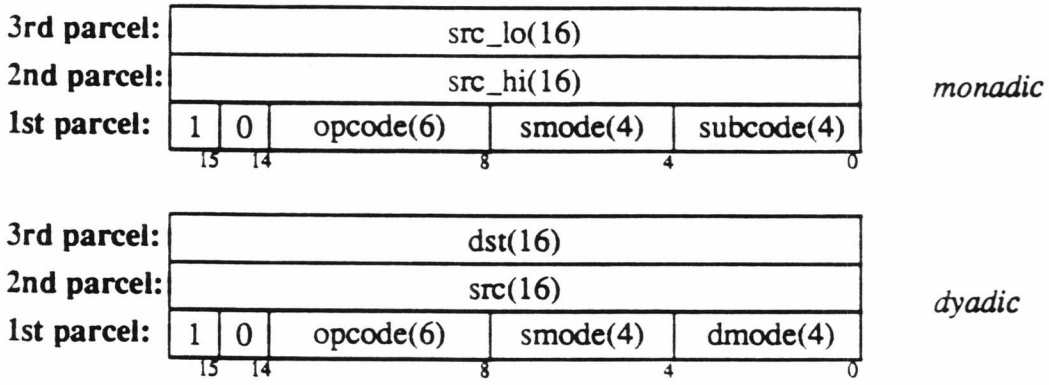
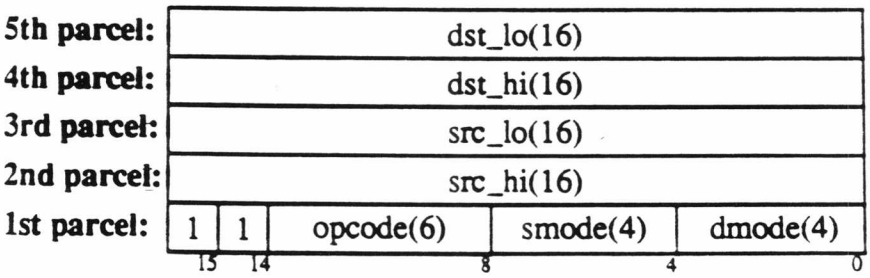


Figure 3c: 10-byte instruction format



AT&T - PROPRIETARY
Use pursuant to Company Instructions

Table 1: Instruction Addressing Modes

These modes are encoded in the "smode" and "dmode" fields of the instruction

Table 1a: General Addressing Mode Encodings

Mode	Code	Description
*$addr:B	0x0	byte absolute
*$addr:UB	0x1	unsigned byte absolute
*$addr:H	0x2	half-word absolute
*$addr:UH	0x3	unsigned half-word absolute
Roffset:B	0x4	byte stack offset
Roffset:UB	0x5	unsigned byte stack offset
Roffset:H	0x6	half-word stack offset
Roffset:UH	0x7	unsigned half-word stack offset
*Roffset:B	0x8	byte stack offset indirect
*Roffset:UB	0x9	unsigned byte stack offset indirect
*Roffset:H	0xA	half-word stack offset indirect
*Roffset:UH	0xB	unsigned half-word stack offset indirect
*$addr:W	0xC	word absolute
Roffset:W	0xD	word stack offset
*Roffset:W	0xE	word stack offset indirect
$data	0xF	immediate

Table 1b: Call/Jump Addressing Mode Encodings

Mode	Code	Description
**$addr	0xC	absolute indirect
*Roffset	0xD	stack offset indirect
Label	0xE	program counter relative
*$addr	0xF	absolute

AT&T - PROPRIETARY
Use pursuant to Company Instructions

Table2: Summary of AT&T Hobbit instruction set and instruction execution times

In the following tables, the minimum time assumes that there is no indirection, and that the data references are in the stack cache. For indirection add 1 cycle if the address is in the stack cache, or memory access time if the operand is not in the stack cache. If the optional [3] is included in the mnemonic, the accumulator is used as the implicit destination.

Table 2a: Integer Arithmetic Instructions

Mnemonic	Description	Min	Max
ADD[3]	add	1	1
ADDI	add interlocked	2	2
DIV[3]	divide	38	38
MUL[3]	multiply	3	20
REM[3]	remainder	38	38
SUB[3]	subtract	1	1
TADD	tagged add	1	1
TSUB	tagged subtract	1	1
UDIV	unsigned divide	38	38
UREM	unsigned remainder	38	38

Table 2b: Logical Instructions

Mnemonic	Description	Min
AND[3]	bitwise logical and	1
ANDI	interlocked AND	2
OR[3]	bitwise logical or	1
ORI	interlocked OR	2
XOR[3]	bitwise logical xor	1

AT&T - PROPRIETARY
Use pursuant to Company Instructions

Table 2c: Shift Instructions

Mnemonic	Description	Min
SHL[3]	left shift	1
SHR[3]	arithmetic right shift	1
USHR[3]	logical right shift	1

Table 2d: Compare Instructions

Mnemonic	Description	Min
CMPEQ	equality comparison	1
CMPGT	signed greater than comparison	1
CMPHI	unsigned greater than comparison	1

Table 2e: Move Instructions

Mnemonic	Description	Min	Max
DQM	double or quad word move	2^a+1	$8*M^b$
LDRAA	load PC-relative address into accumulator	1	
MOV	move	1	
MOVA	move address	1	

a: This is for a constant to stack cache double word DQM
b: M is the access time for one word from off-chip memory

AT&T - PROPRIETARY
Use pursuant to Company Instructions

Table 2f: Program Control Instructions

Mnemonic	Description	Min
CALL	call subroutine	1
CRET	return from kernel with context	$11+Q^a$
JMP	unconditional jump	0^b
JMP(F/T)(Y/N)	conditional jump with prediction[c]	0
KCALL	kernel call	$8+D+A^d$
KRET	return from kernel	10+D
RETURN	free procedure frame and return from subroutine	2^e

a: Q-> memory access time for a quad word
b: If an unconditional jump is folded into the previous instruction, it takes no time to execute, otherwise it takes one cycle.
c: F/T are the branch conditions (false/true), Y/N is the branch prediction (predicted/not predicted)
d: D-> memory access time for a double word, A-> memory access time for a single word.
e: Do not add delay for indirection since RETURNs are always indirect.

Table 2g: Stack Manipulation Instructions

Mnemonic	Description	Min	Max
CATCH	fill the stack cache (if needed)	1	$1+C*Q^a$
ENTER	allocate the procedure frame	1	$1+E*Q^b$
POPN	free N entries from procedure frame	2	2

a: C-> catch amount in quad words, Q-> memory access time for a quad word
b: E-> enter amount in quad words.

AT&T - PROPRIETARY
Use pursuant to Company Instructions

Table 2h: Miscellaneous Instructions

Mnemonic	Description	Min
CPU	register mode escape	0[a]
FLUSHD	flush data cache (unimplemented)	
FLUSHDCE	flush an entry in the data cache (unimplemented)	
FLUSHI	flush the decoded instruction cache	1
FLUSHP	flush the prefetch buffer cache	1
FLUSHPBE	flush an entry in the prefetch buffer cache	1
FLUSHPTE	flush an entry in the TLB	1
NOP	no operation	1
TESTC	copy the carry bit to the flag bit in PSW	1
TESTV	copy the overflow bit to the flag bit in PSW	1

a: CPU instruction is a signal to the prefetch decode unit to interprete the addressing mode of the following instruction differently.

Table 2i: Floating Point Instructions
(Reserved for future implementation)

Mnemonic	Description
FADD[3]	floating-point add
FCLASS	floating-point classification
FCMPEQ	floating-point equality comparison
FCMPEQN	floating-point equal to or unordered with comparison
FCMPGT	floating-point greater than comparison
FCMPGE	floating-point greater than or equal to comparison
FCMPN	floating-point unordered with comparison
FDIV[3]	floating-point division
FLOGB	floating-point exponent extraction
FMOV	floating-point move
FMUL[3]	floating-point multiplication
FNEXT	floating-point next after
FREM	floating-point remainder
FSCALB	floating-point scaling by a power of radix 2
FSQRT	floating-point square root
FSUB[3]	floating-point subtraction

Table 3: CPU register summary

These registers are only writable in kernel mode.

Register Name	Description
MSP	Maximum Stack Pointer (Stack Cache)
SP	Stack Pointer
ISP	Interrupt Stack Pointer
CONFIG	Configuration Register
PSW	Process Status Word
SHAD	Shadow for the Currently Selected Stack Pointer
VB	Vector Base (for event handler vector table)
STB	Segment Table Base (for MMU translations)
FAULT	Fault Register (for MMU page faults)
ID	ID Register (manufacturer encoded)
TIMER1	Timer 1
TIMER2	Timer 2

Table 4: Vector Table Structure

The base of this table is pointed to by the VB register which is quadword aligned.

VB + 0x30	Timer2 interrupt
VB + 0x2C	Timer1 interrupt
VB + 0x28	Interrupt 6
VB + 0x24	Interrupt 5
VB + 0x20	Interrupt 4
VB + 0x1C	Interrupt 3
VB + 0x18	Interrupt 2
VB + 0x14	Interrupt 1
VB + 0x10	NMI
VB + 0x0C	Unimplemented Inst.
VB + 0x08	Niladic Traps
VB + 0x04	Exception Handler
VB + 0x00	KCALL Handler

AT&T - PROPRIETARY
Use pursuant to Company Instructions

Table 5: Processor Pinout Designations

Function	Type	Name	Description	Pins
Start Cycle	O	STC- *	Start of cycle Strobe	1
Read/Write-	O	R/W- *	Read or Write Transfer Strobe	1
Cache	O	CACHE-*	Address may be Cached	1
IO Count	O	IOCOUNT[1:0] *	Block Transfer I/O Remaining	2
Text/Data	O	T/D- *	Text or Data Bus Transaction	1
Bus Lock	O	LOCK- *	Multiple Transfer Bus Lock	1
Address Bus	O	A[31:02] *	Address Bus	30
Byte Marks	O	BM[3:0]- *	Byte Active during I/O Transfer	4
Data Bus	I/O	D[31:00] *	Bi-directional Data Bus	32
Data Ack.	I	DTACK-	Data Transfer Acknowledge	1
Bus Error	I	BERR-	Transaction Error Signal	1
Hold	I	HOLD- *	Bus Transaction Hold	1
Retry	I	RETRY- *	Bus Transaction Retry	1
Bus Grant	I	BGRANT-	Bus Grant	1
Bus Request	O	BREQ-	Processor Bus Request	1
Bus Grant Ack	O	BGACK-	Bus Grant Acknowledge	1
Interrupt Request	I	IL[2:0]	Interrupt Level	3
Clock Stop	I	STOP-	Processor Stop	1
Data Tri-state	I	DTRI-	Data Tri-state	1
Reset Signal	I	RESET-	Reset	1
Clock Input	I	CK23	Phase 23 Clock Input	1
Clock Input	I	CK34	Phase 34 Clock Input	1
Test Clock	I	TCK	Test Clock Input	1
Test Input	I	TDI	Test Data Input	1
Test Mode	I	TMS	Test Mode Select	1
Test Reset	I	TRST-	Test Reset	1
Test Out	O	TDO	Test Data Output	1
Power	P	V_{DD}[18:00]	Power Pins	19
Ground	G	V_{SS}[19:00]	Ground Pins	20

A - indicates that the signal is inverted.
A * indicates the signal is tri-stated if the bus is not owned after deassertion of BGRANT-.

Table 6: Pin Electrical Specifications

Parameter	Symbol	Min	Nom	Max	Unit
Supply Voltage	V_{DD}	3.135	3.3	5.25	V (DC)
Input High Voltage	V_{IH}	V_{DD}-0.5	V_{DD}	V_{DD}+0.5	V (DC)
Input Low Voltage	V_{IL}	-0.5	0	+0.5	V (DC)
TDI Input Low Current	I_{TDI}	-	-	-(0.36+(V_{DD}-3)0.32)	mA
TMS Input Low Current	I_{TMS}	-	-	-(0.16+(V_{DD}-3)0.16)	mA
TCK Input Low Current	I_{TCK}	-	-	-(0.18+(V_{DD}-3)0.16)	mA
TRST- Input High Current	I_{TRST}	-	-	-(0.32+(V_{DD}-3)0.22)	mA
Input Leakage Current	I_I	-1	-	1	µA
Output High Voltage	V_{OH}	V_{DD}-0.25	V_{DD}	V_{DD}+0.25	V (DC)
Output Low Voltage	V_{OL}	-0.25	0	0.25	V (DC)
Output High Current	I_{OH}	-1	-	-	mA
Output Low Current	I_{OL}	-	-	1	mA
Tri-stated Output Leakage Current	I_{OTI}	-1	-	1	µA
20 MHz Supply Current	I_{DD}	-	-	225[a]	mA
Standby Current	I_{SB}	0	-	40[a]	µA

a: at 3.3 V.

Table 7:Timing for Clock Inputs CK23 and CK34
Note that CK23 and CK34 are 90° out of phase

Description	Min (ns)	Nom (ns)	Max (ns)	Var (ns)
Rise Time	-	2.0	3.0	±1.0
Pulse High	23.5	25.0	26.5	±1.5
Fall Time	-	2.0	3.0	±1.0
Pulse Low	23.5	25.0	26.5	±1.5
Period	50.0	-	100.0[a]	-
Delay (between the rising edge of CK23 and the rising edge of CK34)	11.5	12.5	13.5	±1.0

a: Note that by extending the rising edge of CK23 by an arbitrary amount the power saving mode is entered.

Table 8: Absolute Maximum Ratings

Rating	Symbol	Min	Max	Unit
Supply Voltage	V_{DD}	-0.3	+5.5	V (DC)
Ambient Operating Temperature	T_A	0	+70	^{o} C
Junction Operating Temperature	T_J	0	+125	^{o} C
Storage Temperature	T_S	-55	125	^{o} C

AT&T - PROPRIETARY
Use pursuant to Company Instructions

References

1. D. R. Ditzel and H. R. McLellan, "Register Allocation for Free: The C Machine Stack Cache," Proceedings of the Symposium on Architectural Support for Programming Languages and Operating Systems, March 1982.

2. D. D. Hill, "An Analysis of C Machine Support for Other Block-Structured Languages," Computer Architecture News, September 1983

3. A. D. Berenbaum, D. R. Ditzel, and H. R. McLellan, "Introduction to the CRISP Instruction Set Architecture," Spring COMPCON 87 Proceedings, 1987.

4. A. D. Berenbaum, D. R. Ditzel, and H. R. McLellan, "Architectural Innovations in the CRISP microprocessor," Spring COMPCON 87 Proceedings, 1987

5. S. Bandyopadhyay, V. S. Begwani, and R. B. Murray, "Compiling for the CRISP microprocessor," Spring COMPCON 87 Proceedings, 1987.

6. A. D. Berenbaum, D. R. Ditzel, and H. R. McLellan, "The Hardware Architecture of the CRISP Machine," 14th Annual Symposium on Computer Architecture, 1987.

7. D. R. Ditzel and H. R. McLellan, "Branch Folding in the CRISP Microprocessor: Reducing Branch Delay to Zero," 14th Annual Symposium on Computer Architecture, 1987.

AT&T - PROPRIETARY
Use pursuant to Company Instructions